Geo-Targeted Alerting System (GTAS) System Design and Implementation Document

March 18, 2009

NOAA/OAR/ESRL
Global Systems Division
Information Systems Branch

Boulder, CO

Table of Contents

1. Introduction

This document describes the design and implementation of the GTAS system. It will be used by developers during implementation and by management to obtain a measure of the complexity and scope of the system. The system functional requirements are derived from the tasks identified in the MOA (Memorandum Of Agreement) between NOAA and DHS.

The GTAS prototype system will test several new operational and system concepts intended to improve emergency management. The system architecture is specifically designed to support this test and evaluation.

The GTAS prototype takes advantage of several existing technologies. The National Oceanic and Atmospheric Administration (NOAA) in conjunction with other research organizations is developing a very high spatial resolution forecast model, WRF (Weather Research Forecast). This model incorporates terrain and provides a detailed description of the atmosphere at a particular point in time. The WRF model is state-of-the-art and provides its output in the form of a four dimensional (space plus time) grid.

The forecast grid is the input to the dispersion model which determines the motion of gases or other particles in the atmosphere. The dispersion model that is widely used by NOAA is known as HYSPLIT. It was developed by the Air Research Laboratory and can track over five hundred different chemicals and particulates. HYSPLIT can provide its output in several forms for display or post-processing.

A multi-purpose server is used to produce a graphic metafile and distribute it to remote clients for display. This capability comprises a combination of two existing systems for the server and remote client. The Advanced Weather Information system is used by the server since it has access to a diverse set of current NWS weather data and can create the proper metafiles. FSL's Experimental Collaboration (FXC) system is written in Java and has been used by the NWS and other agencies for displaying these metafiles and collaborating among users.

GTAS is an integration of these systems and can be extended to include other dispersion and weather forecast models. The GTAS prototype will help define the system and user requirements for the AWIPS II system currently being developed by Raytheon for the National Weather Service. It will also assist emergency operations centers (EOC) to refine their operational procedures.

2. System Overview

The GTAS system connects FEMA emergency managers with NWS operational forecasters. The system promotes common situational awareness through real-time collaboration among remote users.

The system will use a centralized server that has access to a variety of real-time weather data (including a high resolution forecast model), can run the HYSPLIT dispersion model quickly, and can supports a number of remote collaborative clients. Each client will be able to run the dispersion model independently and collaborate the results with other clients. The emergency manager can manually draw a warning box over the affected area and create a CAP (Common Alert Protocol) message. Depending on the system and network configuration at the office, can then transmit this message to a remote vendor. The basic system concept is illustrated in Figure 1.

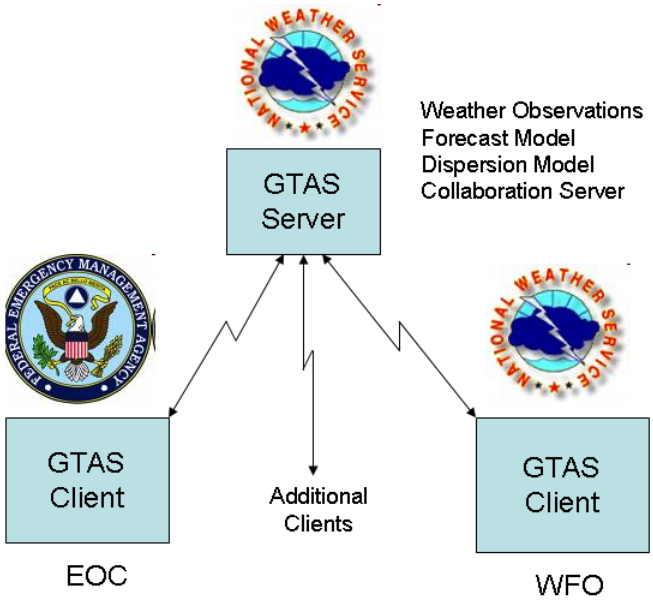


Figure 1 Basic System Architecture

3. Design Approach

3.1 Current Process

The GTAS development will leverage proven existing system technology and integrate it into a functional prototype. Using existing technology will reduce the development risk and improves the chance of fielding a system in the desired time frame.

The system integration process consists of defining and developing the interfaces to major system components and making specific enhancements to the existing code to meet the requirements of the GTAS project. The four major interfaces are:

- WRF to HYSPLIT
- HYSPLIT to AWIPS
- AWIPS to FXC
- FXC to EOC and NWS users

The block diagram (Figure 2) depicts the major system components and these interfaces. Although, each of the system components has some existing external interfaces, additional work is required to have them adhere to the same standards and provide the necessary interface information.

System data flow diagrams and data dictionaries are used to convey the high level flow of information through the system. The data dictionaries will provide the necessary interface specification. The various system components have their own documentation.

3.2 System Alternatives

Currently, most EOCs have separate sytems to view weather data, run a dispersion model, and distribute warnings to vendors. The NWS forecast office is usually contacted by phone for specific local weather information if additional information is needed. The Disaster Management Interoperability Services (DMIS, DM-OPEN) addresses some of the collaboration issues and provides access to some forecast data. However, it does not leverage the extensive NWS infrastructure and field operations. There also exist several commercial products by such companies as MyStateUSA and GeoCast, but these are primarily notification systems and are also proprietary, making it difficult to modify. Other systems, such as the ARAC client, CAMEO, and ALOHA are primarily for producing the dispersion plume. Some of these technologies will be investigated for integration with GTAS. None of these systems appear to use the high resolution data available from WRF.

3.3 Technology Forecasts

The NWS is currently developing a new service-oriented architecture for AWIPS II. This system has the full capabilities of AWIPS I with additional features

planned to support collaboration and remote clients. It is recommended that the GTAS concept be incorporated into this new NWS operational system.

3.4 Design Trade-offs

The following design trade-offs were performed:

Predefined areas vs. dynamic areas – The WRF domain can be specified as nested areas (small high resolution areas within a large low resolution area) or as a single domain that is dynamic in size and location. By using the nested approach, the forecast model can run routinely and the output is always available immediately upon request. This manifests itself as quick response at the user interface. The dynamic area approach requires that the WRF model be run over the specified domain when requested by the user, The model is expected to take about 30 minutes (depending on area, resolution, and setup) to create its output. The nested approach was selected in order to obtain the highest resolution data with little delay. A future enhancement may be a hybrid approach, where a lower resolution model is run over a large area and the high resolution is run on demand.

Centralized vs. local model execution – Weather Forecast Offices acquire a significant amount of local weather observations in addition to that provided by national data centers. Combined with the low cost of very high performance computers, it is possible now for these offices to run fairly sophisticated local models. One of the challenges is to ensure that all offices run the same model with the same data, and initialization in order to have forecast consistency between offices. Also, there is significant cost in purchasing hardware for all of the offices, install the model, and support reliable daily operation. By running the model at a central location (e.g. regional headquarter), these costs can be reduced significantly. Also, this approach provides greater flexibility to support a limited number of designated offices during an evaluation, such as GTAS.

4. Related Documents

WRF Design
ftp://info.mcs.anl.gov/pub/tech_reports/reports/P735.pdf

WRF Development
http://www.ipd.anl.gov/anlpubs/2001/02/38444.pdf

WRF Implementation and Performance
http://wrf-model.org/PRESENTATIONS/UGC_2002/john_michalakes.htm

AWIPS Design
http://fxa.noaa.gov/overview/WFO-A-Overview.book.html

FXC

http://fxc.noaa.gov/fxcdoc/FXC_op_manual.html

5. System Architecture

The four major system components and their interactions are shown in the system block diagram (Figure 2).

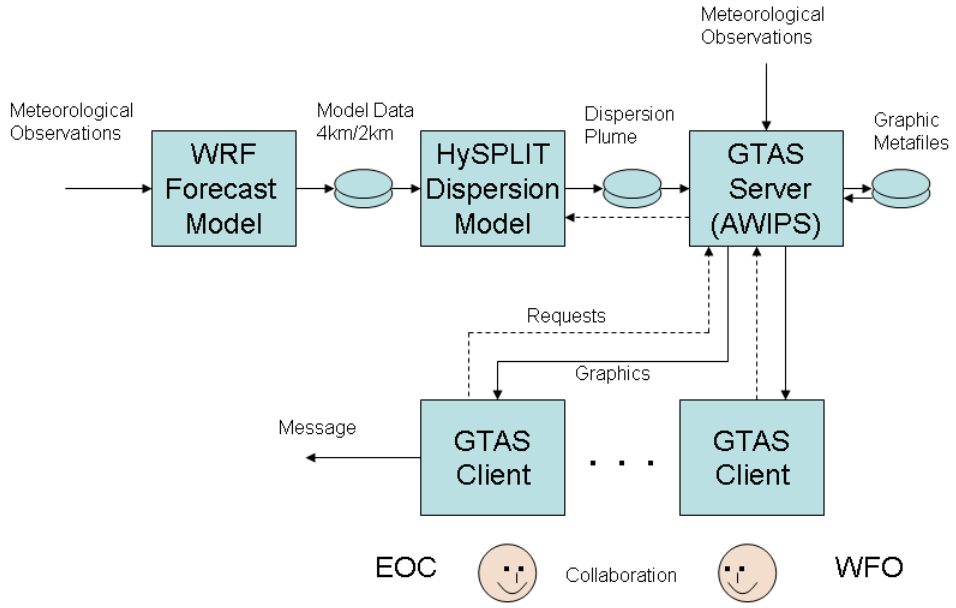


Figure 2 GTAS System Block Diagram

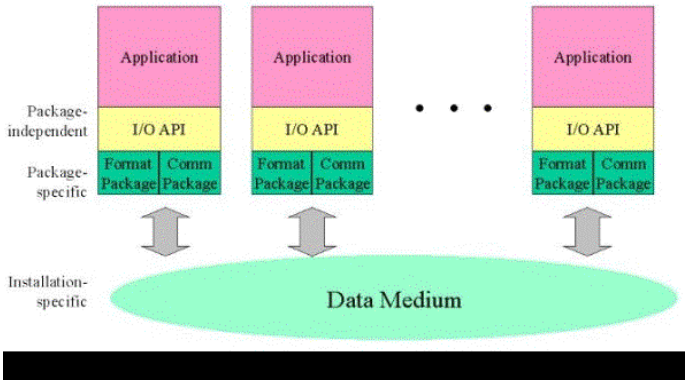
5.1 Software Architecture

The GTAS system consists of four major components, which for the purpose of this document are considered COTS (Commercial Of-The-Shelf) software. However, some modification of each of these components is necessary to properly integrate these systems and provide the necessary functionality.

5.1.1 Weather Research and Forecast Model (WRF)

An introduction to the WRF software architecture, as described by John G. Michalakes[1], Michael McAtee[2], Jerry Wegiel[3] , is stated herein:

The WRF software architecture (Figure 1) consists of three distinct layers: a *model layer* that is usually written by scientists, a *driver layer* that is responsible for parallel decomposition, allocating and deallocating memory, and controlling the integration sequence and I/O, and a *mediation layer* that communicates between the driver and model. In this manner, the user code is isolated from the concerns of parallelism. The design facilitates interchangeable, reusable software; for example other drivers can be used without affecting the underlying model. In like fashion, driver software written for the WRF model can be used for other models, provided that they conform to the interface specifications.



WRF model layer subroutines perform actual computations in the model: advection, diffusion, time-integration, physics, etc. and are required to be *tile-callable*, which means the routine can be called for an arbitrary rectangular subset of the domain. Whereas the driver layer stores all the fields for a domain within a single data object, a Fortran90 derived data-type that is opaque to the driver layer, the model layer deals exclusively with individual data arrays passed as arguments – the mediation layer is responsible for dereferencing the arrays from the derived data types. The interface between the WRF model layer and other layers of the architecture is well defined and public. Adherence to this interface is a requirement for model layer code to be incorporated into the WRF software.

The WRF model layer interface specifies a strong version of the interface rules outlined in Kalnay et al., 1989. The *domain*, *memory*, and *run* dimensions are passed separately and unambiguously into the model layer subroutine as integer arguments denoting starts and ends of each of the three spatial domains (eighteen arguments total). Domain dimensions describe the logical (undecomposed) domain and are used primarily for boundary tests. Memory dimensions provide the dimensions of local memory arrays sized to handle the data and halo regions allocated to a process. The run dimensions provide starts and ends for loops over the local subdomain. All indexing within the WRF model is global; that is, the value of an i, j, or k reference into a state array is always the same – the undecomposed index of the point -- regardless of how the domain is subdivided

over process memories. To ensure thread-safety, there is no communication of model state through common blocks or as module data and there may be no saved data within the subroutine itself. WRF model layer subroutines may not include I/O, interprocessor communication, or multi-threading mechanisms; therefore, the temporal scope of a model layer subroutine is that unit of computation that can be performed without concern for coherency – that is, without the need to communicate (e.g. update halo regions) with other processes or synchronize with other threads.

This layered approach to the WRF software design, and in particular the model-layer interface specification, supports a completely general but efficient approach to parallelism by insulating the model layer code from all computer architecture-specific concerns. By following the interface convention, a scientist can contribute code and be assured that it will run in parallel in WRF without any special effort on his or her part. Further, adherence to the rules for plug-compatible physics allows WRF model layer routines to be easily interchanged with other models. The WRF driver layer software can be separated from the WRF model layer and used as a framework to parallelize other components of the WRF system (WRF 3DVAR; see McAtee et al., 2002 in this volume) or other models entirely (the NOAA/NCEP MesoNH model). Finally, the layered design allows WRF to take advantage of existing software infrastructure if it exists at a particular institution -- for example, NOAA/GFDL's Flexible Modeling System (FMS) (GFDL, 2002) – and to leverage broad-based community efforts to develop common model infrastructure such as the NASA-funded Earth System Modeling Framework project (Dickenson et al., 2002).

5.1.2 Hybrid Single-Particle Lagrangian Integrated Trajectory Model (HYSPLIT)

5.1.3 Advanced Weather Information Processing System (AWIPS)

The AWIPS system consists of two major components: the data acquisition and the interactive display component. All real-time data required by the weather forecaster is acquired and processed by the data acquisition component. The largest volume of data is received from the Satellite Broadcast System (SBN) which includes forecast model data, observations, and forecaster guidance products. In addition to the SBN data sets, the system also acquires radar data, local observations, and other data requested by the user. All of this data is encoded or compressed and requires preprocessing to store it into the AWIPS data base in the proper format. Since data decoding can be time consuming, the data controller may initiate several parallel decoders in order to keep up with the incoming data. Once the data has been decoded and stored on the AWIPS data base, a notification is sent to all registered users indicating that the data is now available for additional processing.

The interactive display component consists of five distinct modules which execute as system processes Figure (tbd). These modules are data access (notification), data selection and interaction, data visualization, external functions, and display. The data selection and interaction module provides the user interface and leverages existing toolkits (tcl/tk) for developing the graphical user interface. The core component is the visualization component (IGC) which converts raw data for display according to various user and system inputs. It is the most complex module since it can create a variety of displays as images and vector graphics, and allows the user to specify these displays in an interactive manner. The IGC is a tightly coupled module in order to provide the necessary processing flexibility and quick response. External functions, sometimes referred to as extensions or applications, provide additional functionality which are not part of the core functions. These can be written by users, but require knowledge of the AWIPS software architecture.

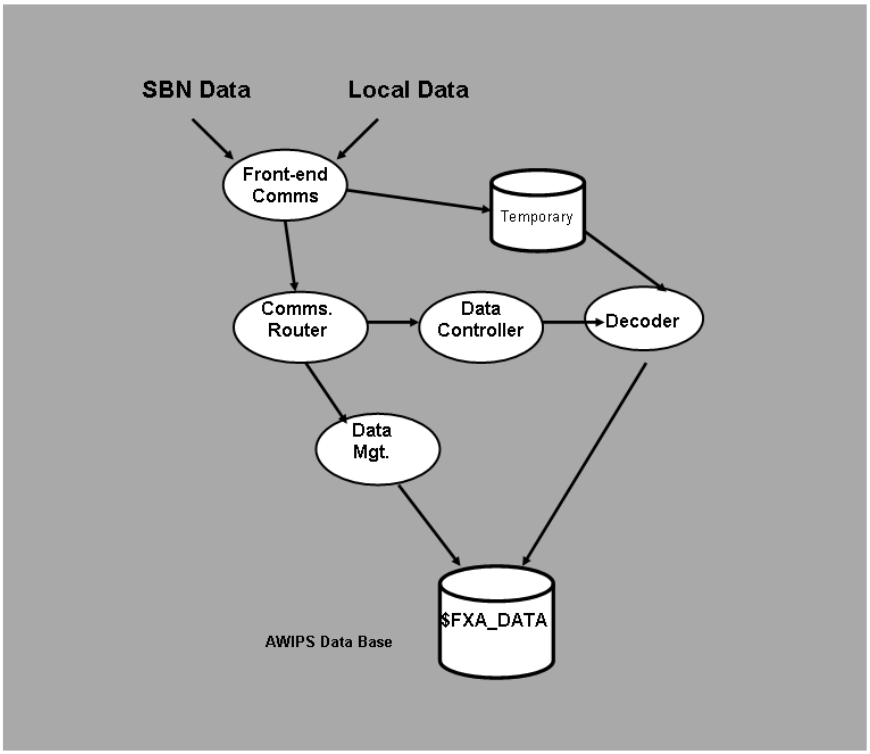


Figure 3 AWIPS Data Acquisition Architecture

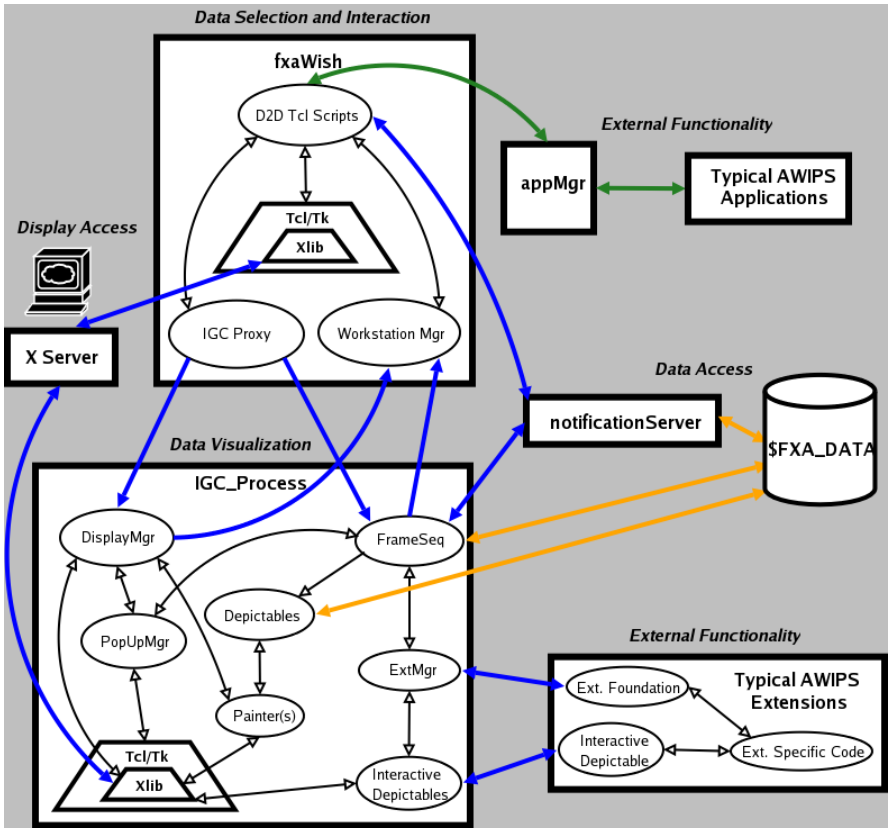


Figure 4 AWIPS Display Architecture

5.1.4 FSL's Experimental Collaboration System (FXC)

The FXC system consists of a client which provides the interface to the user, and a server which provides access to various data.

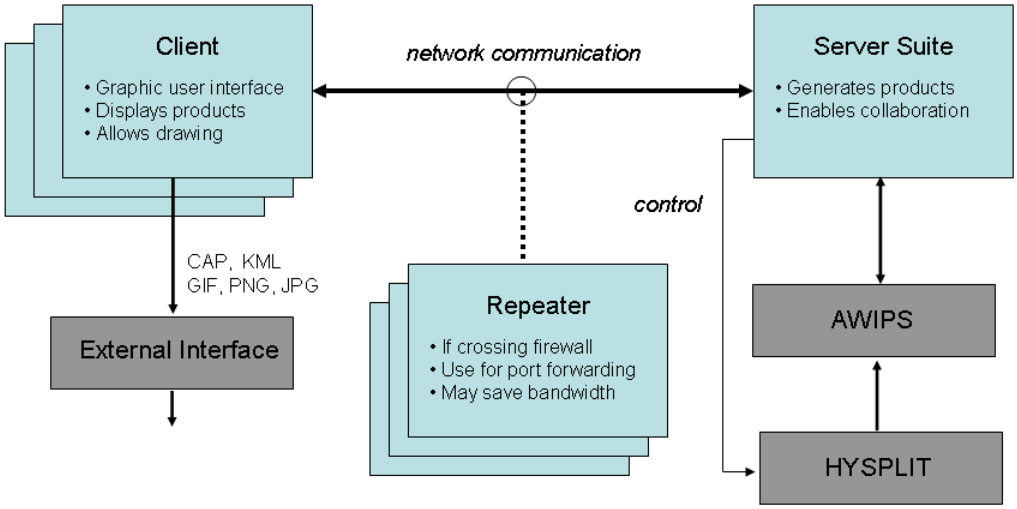


Figure 5 FXC High Level System Architecture

5.2 Hardware Architecture

HPC (High Performance Computer) compute cluster at GSD (phase 1)
Cluster machine at NWS Regions (phase 2)
Dell servers for HySPLIT and AWIPS
Standard PC (Windows or Linux) for client

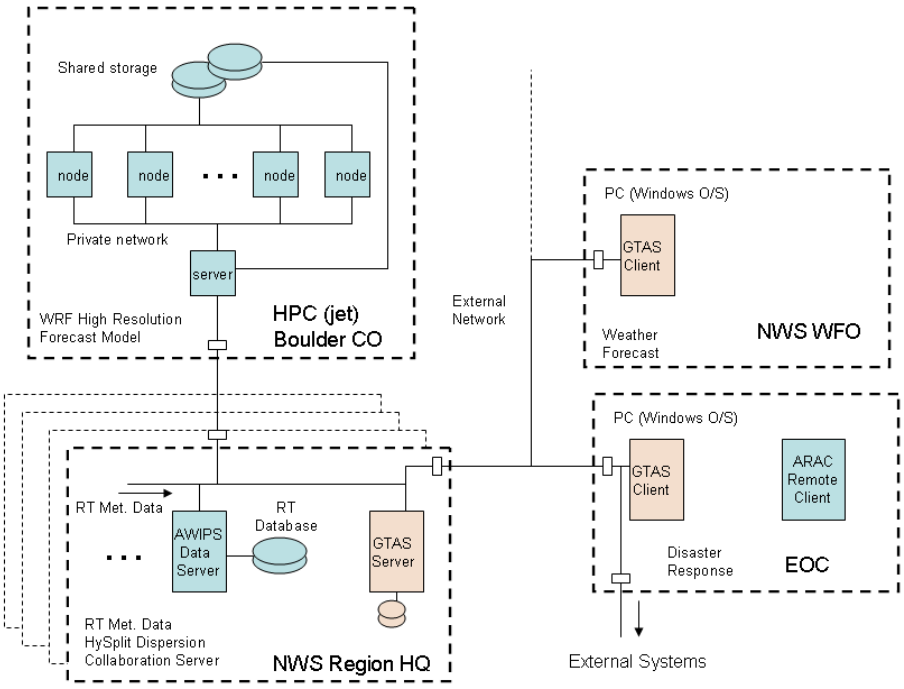


Figure 6 GTAS System Architecture (Phase 1)

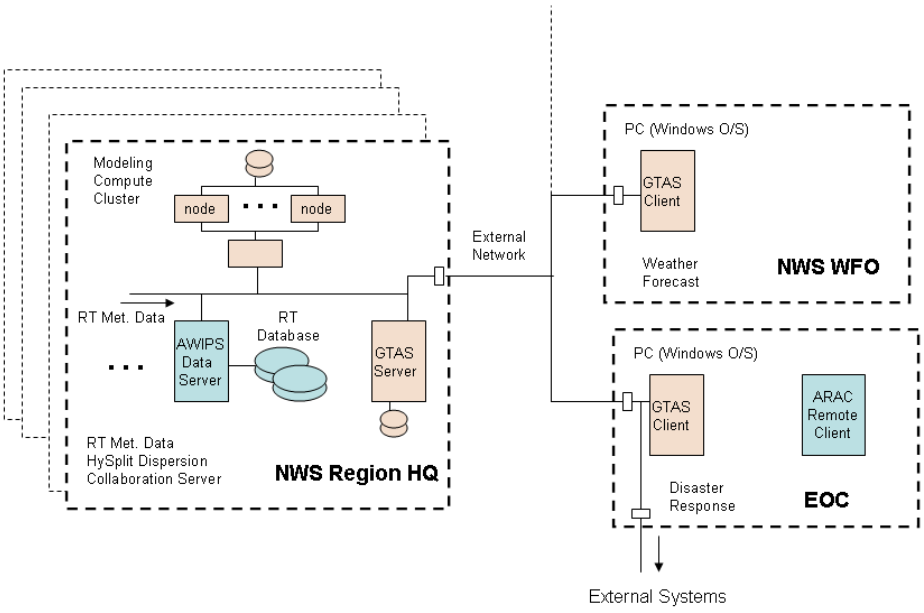
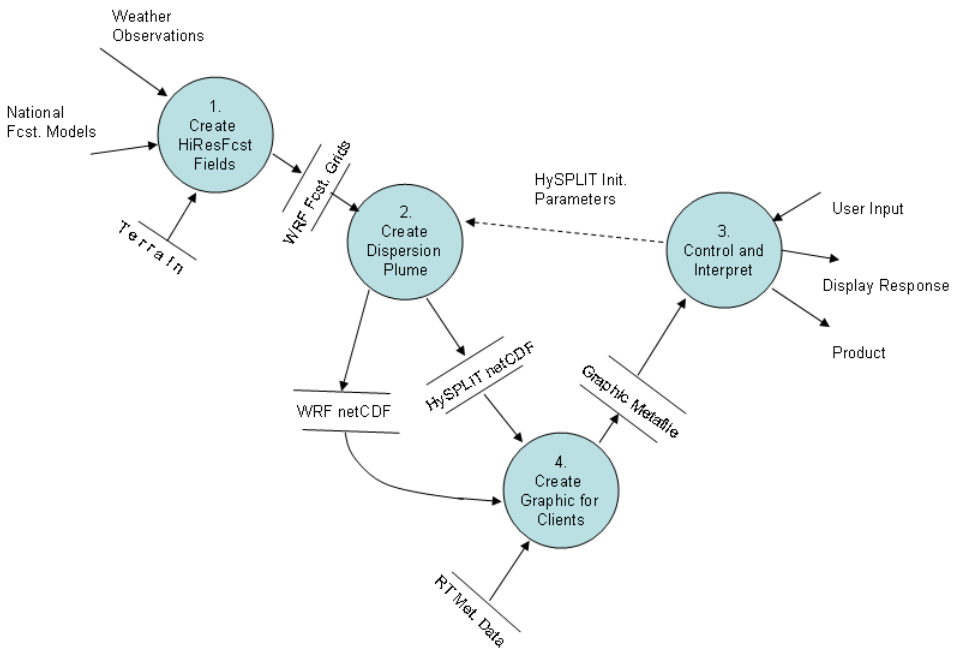


Figure 7 GTAS System Architecture (Phase 2)

5.3 Communications Architecture

6. System Interface Design

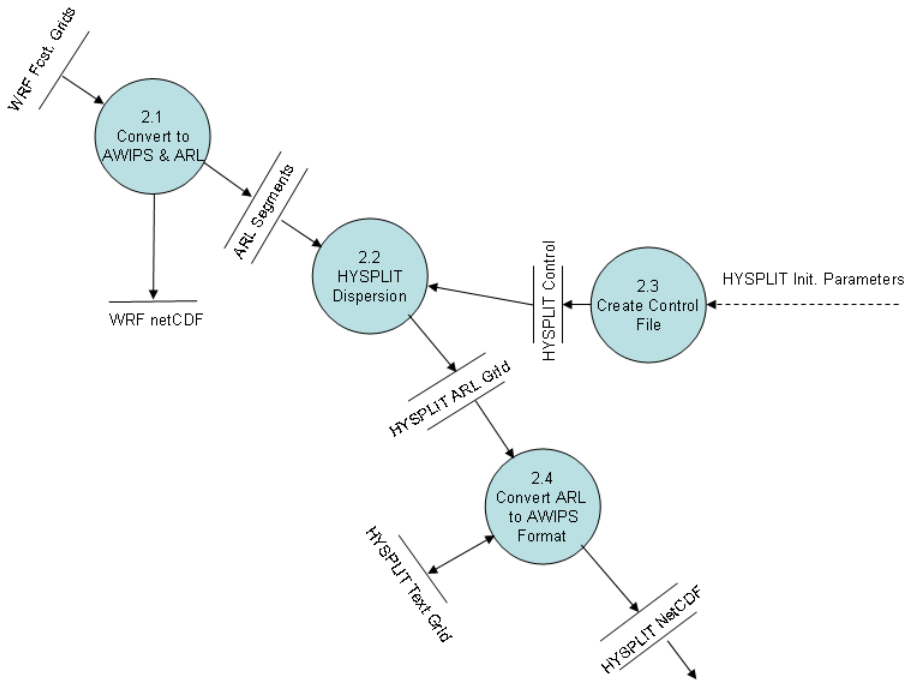
The data flow diagrams are used to identify the processing modules and also the data that has to be passed between the modules. Only the top level and one additional level of detail is shown in the following data flow diagrams.



GTAS Top Level Data Flow

Data Dictionary:

HySplit Init Parameterts = user + site + data type + [start location + start height] + run time + output type + [name + rate + emission time]



2. Create Dispersion Plume DF

6.1 WRF to HYSPLIT Interface

6.3.1 Interface Architecture

6.3.2 Interface Detailed Design

6.2 HYSPLIT to AWIPS Interface

6.3.1 Interface Architecture

6.3.2 Interface Detailed Design

6.3 AWIPS to FXC Interface

6.3.1 Interface Architecture

6.3.1 Interface Detailed Design

6.4 FXC to HYSPLIT Interface

6.4.1 Interface Architecture

6.4.2 Interface Detailed Design

The GTAS client communicates with a “ScriptServer” process (written in Java) on the server which then starts the actual shell script that launches the HYSPLIT model. The user provides the initialization parameters through the graphical user interface and sends them to the ScriptServer by depressing a “run” button. The following control options are available:

runConcNoCtrl.csh <user> <site> <datatype> -t -lh -in -r -ty –p

user -- Model runer's user name
site -- Site can be a region, wfo or city in the region.
datatype -- can be "laps" or "nam12" currently.

-t -- start time,
format: -t yy mm dd hh MM
"-t 0 0 0 0 0" is the current time, the input init Time
example: "-t 09 03 15 23 56"

-lh -- Is the start location(s) and hight(s), the format is,
-lh startnum lat1 long1 high1 ... latN longN highN
for example,
-lh 2 32.0 -104.1 10.0 33.3 -106.6 50

-r -- Total run time, the duration of the calculation in hours.
for example, -r 48

-ty -- model output type value can be "average" or "max"
format: -ty type
example: -ty average

-p -- Pollutant list. Format,
-p pnum name1 rate1 hour1 name2 rate2 hour2... nameN rateN hourN
rate:emission rate(1/hr)
hour:hour of emission
example: -p 2 pol1 1.0 1.0 pol2 1.0 2.0

7. User Interface

7.1 Interface Architecture

7.2 Interface Detailed Design

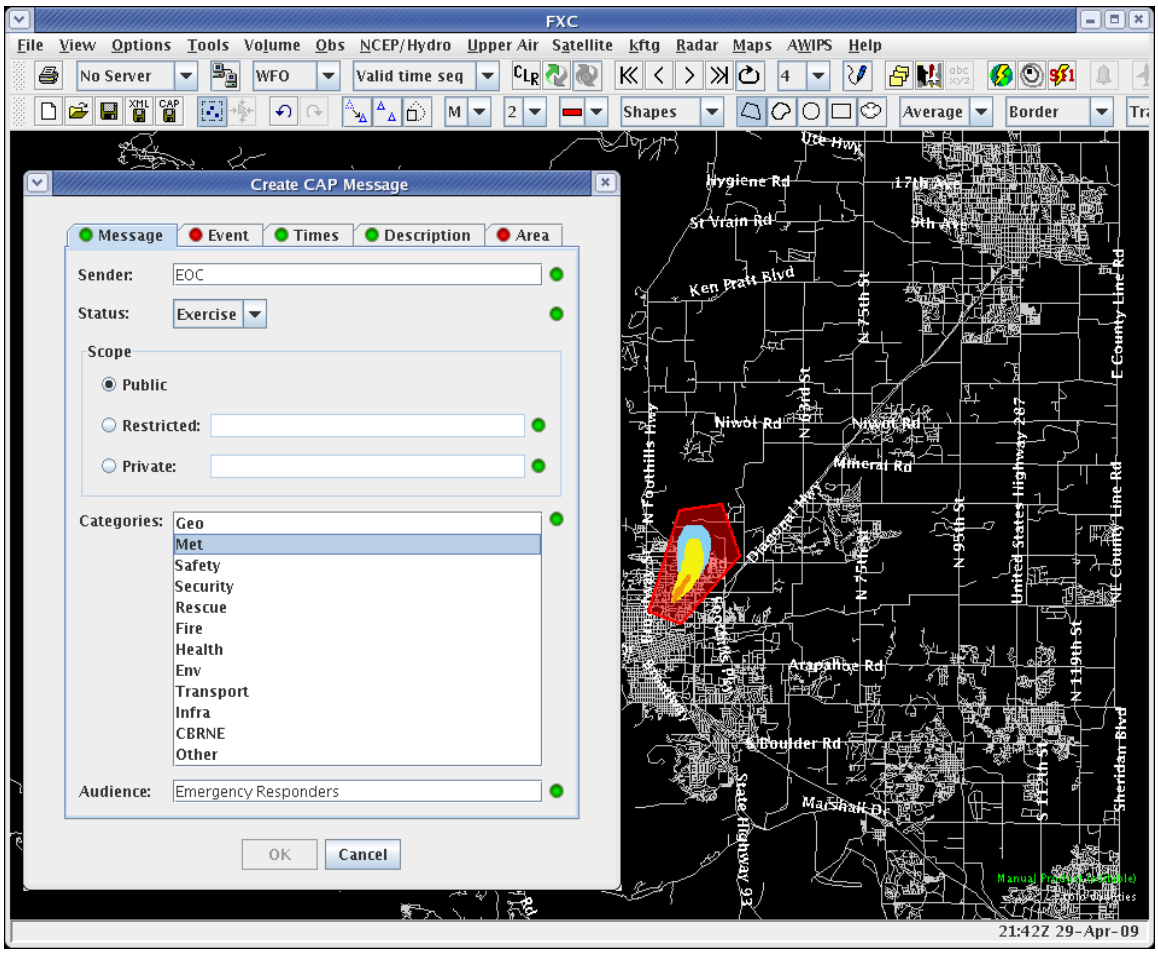


Figure X GTAS menu for creating a CAP message

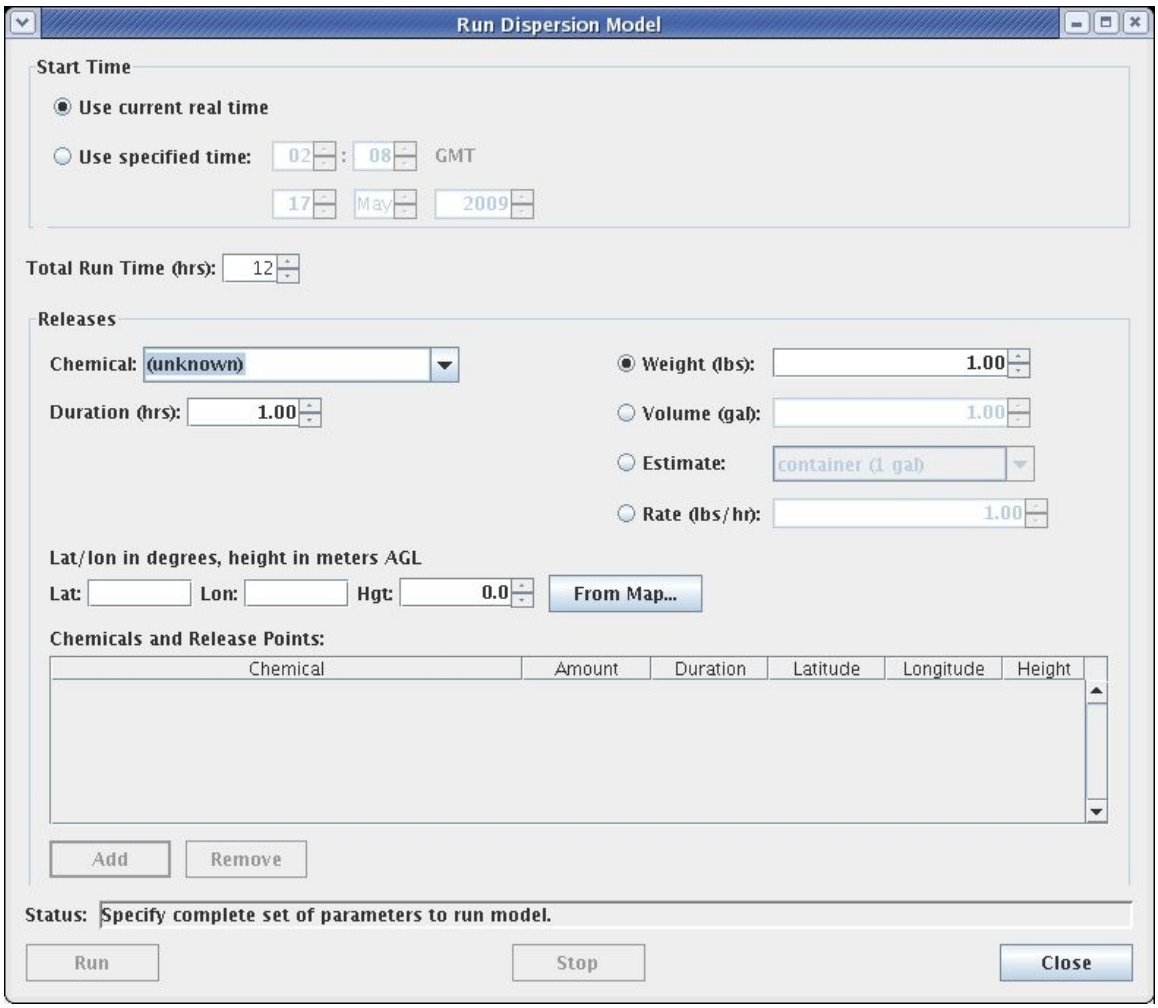


Figure X GTAS menu for starting the HYSPLIT model

7.2.1 Emergency Operations Center

7.2.2 Weather Forecast Office